CLAIMS

What is claimed is:

1. A method of debugging an executing service on a pipelined CPU architecture, the method comprising:

setting a breakpoint within an executing service;

saving a minimum state of the executing service;

altering a program counter of the executing service;

restoring the program counter of the executing service; and

restoring the state of the executing service.

2. The method of claim 1 further comprising:

executing debug commands within the executing service.

3. The method of claim 1 wherein setting the breakpoint further comprises:

locating an original instruction within the executing service to set the breakpoint;

inserting a breakpoint instruction at the breakpoint;

starting the executing service;

waiting for the breakpoint to execute;

waiting for memory fetches and configuration loads to complete; and

restoring the original instruction at the breakpoint location.

4. The method of claim 1 wherein setting the breakpoint comprises:

altering an instruction within the executing service at a breakpoint location; and

invalidating a page cache of the executing service.

5. The method of claim 1 wherein setting the breakpoint comprises:

setting a breakpoint register to point to a breakpoint location.

6. The method of claim 1 wherein saving a minimum state comprises:

saving the executing service registers; and

flushing a pipeline of the executing service.

7. The method of claim 6 wherein flushing the pipeline further comprises:

determining if registers are unstable;

if registers are unstable, saving the value of any registers that change after each pipeline cycle; and

if the breakpoint location is set on a location that uses old values of registers, saving the old values of the registers before new values are written to the registers.

8. The method of claim 7 wherein registers are scalar registers or predicate registers.

9. The method of claim 1 wherein altering the program counter further comprises:

setting the program counter of the executing service to point to a save stub;

starting execution of the executing service;

executing the breakpoint;

storing configuration registers of the executing service;

saving values of scalar and predicate registers;

saving pipeline registers; and

storing a stack pointer value for a breakpoint location.

10. The method of claim 1 wherein restoring the program counter further comprises:

setting the program counter of the executing service to point to a restore stub; and

starting the executing service at the breakpoint.

11. The method of claim 1 wherein restoring the state further comprises:

if a breakpoint location is on an instruction that does not make use of old values,

restoring stable registers;

if the breakpoint location is on an instruction that does make use of old values,

restoring unstable registers, and

reloading the pipeline;

altering the program counter of the executing service to point to the breakpoint location; and

starting execution of the executing service at the breakpoint location.

12. The method of claim 1 further comprising:

fetching a page of memory of the executing service into an instruction cache;

checking for a checksum error within the page of memory; and

if the executing service is set to reject the checksum error,

saving the page of memory,

inserting a breakpoint into the saved page of memory,

altering an instruction pointer to the saved page of memory, and

processing the saved page of memory.

13. A method of debugging an executing service on a pipelined CPU architecture, the method comprising:

setting a breakpoint at a last safe point;

saving a minimum state of the executing service;

simulating instructions of the executing service from the last safe point to the breakpoint;

executing debug commands within the executing service; and

restoring the state of the executing service.

14. The method of claim 13 wherein restoring further comprises:

storing the simulated state of the executing server to the CPU; and restoring an original execution.

15. The method of claim 13 wherein simulating further comprises single stepping through a set of unsafe instructions, the set of unsafe instructions are between the last safe point and a next safe point.

16. A method of debugging an executing service on a pipelined CPU architecture without hardware interlocks, the method comprising:

fetching a page of memory of the executing service into an instruction cache;

checking for a checksum error within the page of memory; and

if the executing service is set to reject the checksum error,

saving the page of memory,

inserting a breakpoint into the saved page of memory,

altering an instruction pointer to the saved page of memory, and

processing the saved page of memory.

17. The method of claim 16 wherein processing further comprises:

setting a breakpoint within an executing service;

saving a minimum state of the executing service;

altering a program counter of the executing service;

executing debug commands within the executing service;

restoring the program counter of the executing service; and

restoring the state of the executing service.

18. A system for debugging an executing service on a pipelined CPU architecture without hardware interlocks, the system comprising:

a debugger to set a breakpoint within an executing service and execute debug commands within the executing service;

a save stub to save a minimum state of the executing service and alter a program counter of the executing service;

a processing engine to execute the breakpoint; and

a restore stub to restore the state of the executing service.

19. The system of claim 18 wherein the debugger is further operable to locate an original instruction within the executing service to set the breakpoint, insert a breakpoint instruction at the breakpoint, start the executing service, wait for the breakpoint to execute, wait for memory fetches and configuration loads to complete, and restore the original instruction at the breakpoint location.

20. The system of claim 18 wherein the debugger is further operable to alter an instruction within the executing service at a breakpoint location, and invalidate a page cache of the executing service.

21. The system of claim 18 wherein the debugger is further operable to set a breakpoint register to point to a breakpoint location.

22. The system of claim 18 wherein the save stub is further operable to save the executing service registers.

23. The system of claim 18 wherein the processing engine is further operable to flush a pipeline of a set of pipeline instructions of the executing service.

24. The system of claim 22 wherein the debugger is further operable to determine if registers are unstable, save the value of any registers that change after each pipeline cycle if registers are unstable, save the old values of the registers before new values are written to the registers, and if the breakpoint location is set on a location that uses old values of registers.

25. The method of claim 24 wherein registers are scalar registers or predicate registers.

26. The system of claim 18 wherein the debugger is further operable to set the program counter of the executing service to point to a save stub, start execution of the executing service, execute the breakpoint, store configuration registers of the executing service, save values of the scalar and predicate registers, and save pipeline registers.

27. The system of claim 18 wherein the debugger is further operable to set the program counter of the executing service to point to a restore stub, and start the executing service at the breakpoint.

28. The system of claim 18 wherein the restore stub is further operable to:

if a breakpoint location is on an instruction that does not make use of old values,

restore stable registers;

if the breakpoint location is on an instruction that does make use of old values,

restore unstable registers, and

reload the pipeline;

alter the program counter of the executing service to point to the breakpoint location; and

start execution of the executing service at the breakpoint location.

29. The system of claim 28 wherein the restore stub is further operable to reload the pipeline state directly.

30. The system of claim 28 wherein the restore stub is further operable to re-execute the original instructions within the pipeline to recreate the pipeline at a time of the breakpoint.

31. The system of claim 18 wherein the processing engine is further operable to:

fetch a page of memory of the executing service into an instruction cache; and

check for a checksum error within the page of memory.

32. The system of claim 18 wherein the debugger is further operable to:

if the executing service is set to reject the checksum error,

save the page of memory,

insert a breakpoint into the saved page of memory,

alter an instruction pointer to the saved page of memory, and

process the saved page of memory.

33. A system for debugging an executing service on a pipelined CPU architecture, the system comprising:

a save stub to save a minimum state of the executing service;

a restore stub to restore the state of the executing service; and

a debugger to set a breakpoint at a last safe point, simulate instructions of the executing service from the last safe point to the breakpoint, and execute debug commands within the executing service.

34. The system of claim 33 wherein the restore stub further stores the simulated state of the executing service to the CPU, and resumes an original execution.

35. The system of claim 33 wherein the debugger is further operable to single step through a set of unsafe instructions, the set of unsafe instructions are between the last safe point and a next safe point.

36. A system for debugging an executing service on a pipelined CPU architecture, the system comprising:

a processing engine to fetch a page of memory of the executing service into an instruction cache, and check for a checksum error within the page of memory; and

if the executing service is set to reject the checksum error, a debugger operable to save the page of memory, insert a breakpoint into the saved page of memory, alter an instruction pointer to the saved page of memory, and process the saved page of memory.

37. The system of claim 36 wherein the debugger is further operable to set a breakpoint within an executing service, save a minimum state of the executing service, alter a program counter of the executing service, and execute debug commands within the executing service.

38. The system of claim 37 further comprising:

a restore stub operable to restore the program counter of the executing service, and restore the state of the executing service.

39. A system for debugging an executing service on a pipelined CPU architecture, the system comprising:

means for setting a breakpoint within an executing service;

means for saving a minimum state of the executing service;

means for altering a program counter of the executing service;

means for restoring the program counter of the executing service; and

means for restoring the state of the executing service.

40. A system for debugging an executing service on a pipelined CPU architecture, the system comprising:

means for setting a breakpoint at a last safe point;

means for saving a minimum state of the executing service;

means for simulating instructions of the executing service from the last safe point to the breakpoint;

means for executing debug commands within the executing service; and

means for restoring the state of the executing service.

41. A system for debugging an executing service on a pipelined CPU architecture, the system comprising:

means for fetching a page of memory of the executing service into an instruction cache;

means for checking for a checksum error within the page of memory; and

if the executing service is set to reject the checksum error,

means for saving the page of memory,

means for inserting a breakpoint into the saved page of memory,

means for altering an instruction pointer to the saved page of memory, and

means for processing the saved page of memory.

42. A computer readable medium comprising instructions, which when executed on a processor, perform a method for debugging an executing service on a pipelined CPU architecture, comprising:

setting a breakpoint within an executing service;

saving a minimum state of the executing service;

altering a program counter of the executing service;

restoring the program counter of the executing service; and

restoring the state of the executing service.

43. A computer readable medium comprising instructions, which when executed on a processor, perform a method for debugging an executing service on a pipelined CPU architecture, comprising:

setting a breakpoint at a last safe point;

saving a minimum state of the executing service;

simulating instructions of the executing service from the last safe point to the breakpoint;

executing debug commands within the executing service; and

restoring the state of the executing service.

44. A computer readable medium comprising instructions, which when executed on a processor, perform a method for debugging an executing service on a pipelined CPU architecture, comprising:

fetching a page of memory of the executing service into an instruction cache;

checking for a checksum error within the page of memory; and

if the executing service is set to reject the checksum error,

saving the page of memory,

inserting a breakpoint into the saved page of memory,

altering an instruction pointer to the saved page of memory, and

processing the saved page of memory.